ln – make links and symlinks between files or directories

ln command is a Unix command for linking files or directories to each other. Essentially, it creates new files with the names you specify, and refer them to already existing files or directories. When you run any Unix command against a symlink, it is first resolved (the original file it points to is confirmed) and the Unix command works with that file to produce desired outcome.

Two types of linking files and directories

There are two common approaches to link a file or directory in Unix: soft linking and hard linking. Soft links are also called symlinks (symbolic links).

What is a soft link?

Soft link *(also referred to as* ***symlink*** – short for symbolic link) is a special type of file in Unix, which references another file or directory. Symlink contains the name for another file and contains no actual data. To most commands, symlinks look like a regular file, but all the operations (like reading from a file) are referred to the file the symlink points to.

When you remove a soft link, you simply remove one of the pointers to the real file. When you remove the original file a soft link points to, your data is lost. Even though your soft link will still exist, it will be pointing to the non-existent file and will therefore be useless (it will probably have to be removed as well).

What is a hard link?

Hard link is a pointer to physical data. Effectively, all standard files are hard links, because they ultimately create an association between a file name and a physical data which corresponds to each file.

In Unix, you can create as many hard links to a file as you like, and there is even a special counter for such references. When you're using the long format of an ls command, you can see this counter.

When you remove a hard link, you decrease this link counter for a data on your storage. If you remove the original file, the data will not be lost as long as there's at least one hard link pointing to it.

Creating soft links (symlinks) with ln

Let's start with a really simple example. We create a text file, and then use soft link to reference it.

This shows how the file is created. It's called **file1**, and has a line of text data in it which we confirm using cat command:

```
ubuntu$ echo "Text file #1" > file1
ubuntu$ cat file1
Text file #1
```

Now let's use **ln** command to create a soft link. The newly created symlink will be a file called **file2**, and **ls** command will show you that it points to **file1**:

```
ubuntu$ ls -l file2
lrwxrwxrwx 1 root root 5 Mar 31 03:54 file2 -> file1
```

If you try accessing the **file2**, you will ultimately access **file1**, that's why the following example shows you the contents of **file1**:

```
ubuntu$ cat file2
```

```
Text file #1
```

Now, if you remove file1, this will make file2 symlink invalid, and any attempts to use it will return a “file not found” type of error:

```
ubuntu$ rm file1
ubuntu$ cat file2
cat: file2: No such file or directory
```

Creating hard links with ln

Now, let’s look at creation of hard links in Unix. For this example, we’ll recreate the **file1**:

```
ubuntu$ echo "Text file #1" > file1
ubuntu$ cat file1
Text file #1
```

If you use ls to look at **file1**, you can see that the link counter (second field from the left) is set to 1 – which means that there is only one file name pointing to the data with our “Text file #1” text:

```
ubuntu$ ls -l file1
-rw-r--r-- 1 root root 13 Mar 31 06:18 file1
```

And now we use **ln** command to create a hard link called **file3**, which points to the same data as **file1**:

```
ubuntu$ ln file1 file3
```

If we use ls command once again, you can see that the link counter has been increased and is now 2:

```
ubuntu$ ls -l file1 file3
-rw-r--r-- 2 root root 13 Mar 31 06:18 file1
-rw-r--r-- 2 root root 13 Mar 31 06:18 file3
```

Notice, how the file1 and file3 files look like absolutely normal files, and there’s nothing showing a logical link between them.

To confirm that both filenames are actually referring to the same area on the disk, you can use -i option for the ls command, which will show you an **i-node** value.

i-nodes are data structures of a filesystem used to store all the important properties of each file: size, owner's user id and group id, access permission and more. The important thing is that each named data area on your disk must have an inode, and when you create a new data file this means creating an i-node. But when you're using hard links, you're effectively creating filesystem directory entry, which references an already existing data, so the hard link gets the same i-node number pointing to the same data.

```
ubuntu$ ls -il file1 file3
655566 -rw-r--r-- 2 root root 13 Mar 31 06:18 file1
655566 -rw-r--r-- 2 root root 13 Mar 31 06:18 file3
```

The first number in each line of the output is the i-node number, and since we're referencing the same data, the i-node numbers are also the same.

See also:

- Unix symlink example
- hard-link in unix
- ln command
- Show what symlink points to